


Domain-Specific Software Architectures (DSSA)


1



Presentation Goals

- Understand Domain Specific Architecture
 - What they are
 - How they help development
- Look at 2 examples

2




What Is DSSA?

- Assemblage of software components
 - Specialized for particular type of task (domain)
 - Generalized for effective use across that domain
 - Composed in standardized structure (topology) effective for building successful applications

» Rick Hayes-Roth, 1994

3

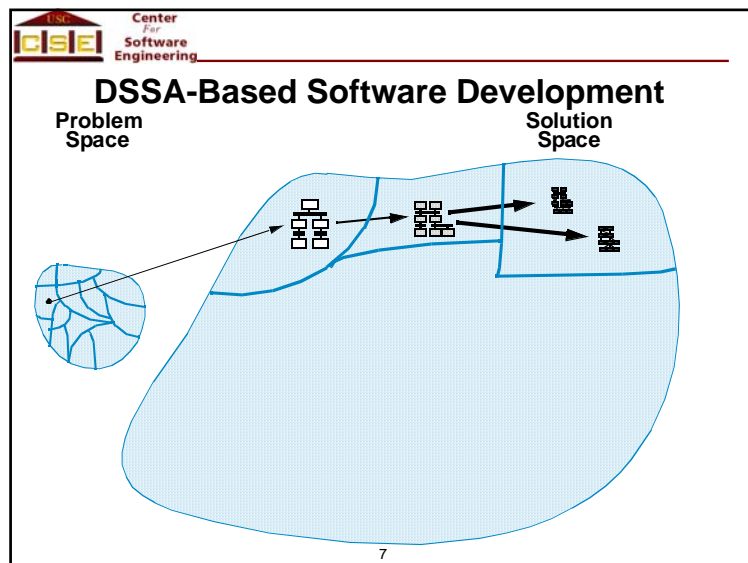
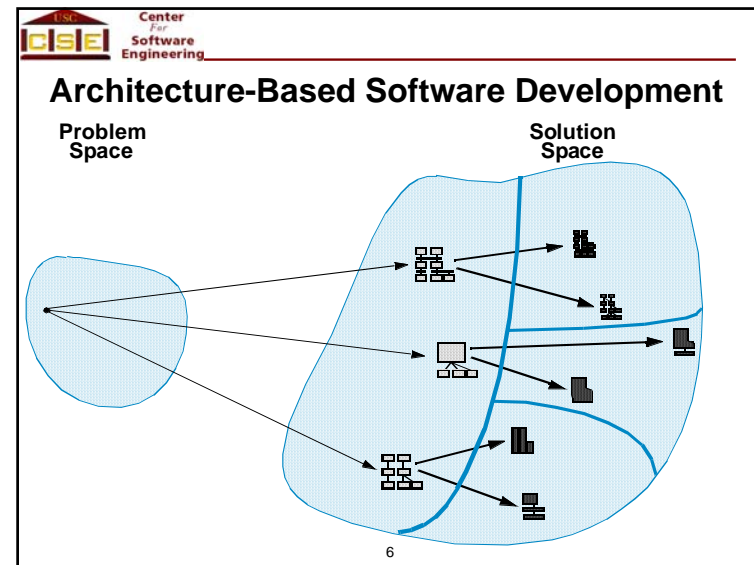
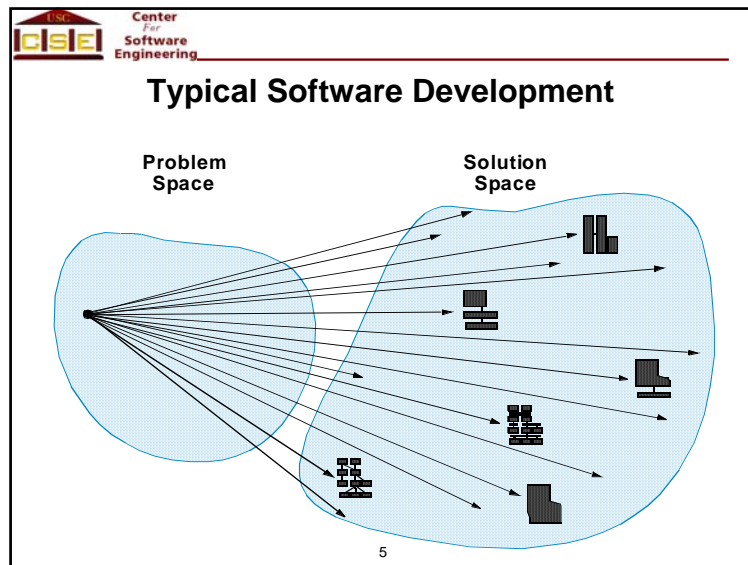


Components of DSSA

- Domain model
- Reference requirements
- Reference (parameterized) architecture
 - Standardized architecture describing all systems in domain
 - Focuses on fundamental domain abstractions
 - Expressed in ADL
- Supporting infrastructure/environment
- Process/methodology to instantiate, refine, & evaluate it

» Will Tracz, 1995


4



Why Domain-specific Architecture?

- Development can be optimized
 - Domain-specific software patterns
 - Domain-specific models & methods to analysis
 - Estimate characteristics
 - Verify product characteristics
- Reuse potential is maximized
 - Domain-specific requirements
 - Domain-specific components
 - Domain-specific software patterns


8

 Center For Software Engineering

What Is A Domain Model?

- Represents domain (problem space)
 - Functions being performed
 - Entities
 - Information
- "All models are wrong; some are useful."
 - » Unknown, SEI
- Fundamental objective:
 - Standardize problem-domain terminology & semantics
 - Terminology + semantics = ontology
 - Basis for standard descriptions of problems to be solved


9

 Center For Software Engineering

Domain Analysis

- Identify, describe, & organizing objects, operations and patterns
 - Described using standardized vocabulary
 - Abstracted to suit
 - Class of similar systems
 - Particular problem domain
 - To be reused when creating new systems
- Produces domain model
- "Domain analysis is like several blind men describing an elephant"


10

 Center For Software Engineering

Elements Of Domain Model (1)

- Customer needs statement
 - Identifies functional requirements
 - Informal
 - High-level
 - Ambiguous
 - Incomplete
- Scenarios
 - Functional requirements
 - Data flow
 - Control flow
 - Elicited from the customer
- Domain dictionary
 - Definitions of terms used
 - Updated over time


11

 Center For Software Engineering

Elements Of Domain Model (2)

- Context (block) diagram
 - High-level connection between major components of system
- Entity-relationship/Class models
 - Aggregation ("a-part-of") & generalization ("is-a") relations
- Data-/Message-flow models
- State transition models
- Object model
 - First phase of component interface design


12



What Are Reference Requirements?

- Requirements that apply to entire domain
- Contain
 - Defining characteristics of problem space
 - Functional requirements
 - Non-functional (Qualities) requirements
 - Limiting characteristics (constraints) on solution space
 - Non-functional requirements (e.g., security, performance)
 - Design requirements (e.g., architectural style, UI style)
 - Implementation requirements (e.g., platform, language)


13



What Is Reference Architecture?

- Standardized, generic architecture describing “all” systems in domain
 - Focuses on fundamental abstractions
 - Expose a hierarchical, compositional nature
 - Syntax & semantics of constituent high-level components are specified
 - Satisfies reference requirements
 - May need set of reference architectures to satisfy all reference requirements
 - Reusable, extendable, & configurable
- Instantiated to create design architecture for specific system


14



Elements Of Reference Architecture

- Reference architecture model
 - Topology based on architectural style
 - Component dependency diagram
 - Component interface descriptions
 - Constraints
 - Parameter value ranges & relationships
- Architecture schema or design record
 - Information about components & design alternatives
 - Domain- & implementation-specific
- Rationale
- Configuration decision tree
 - Used to instantiate system from reference architecture

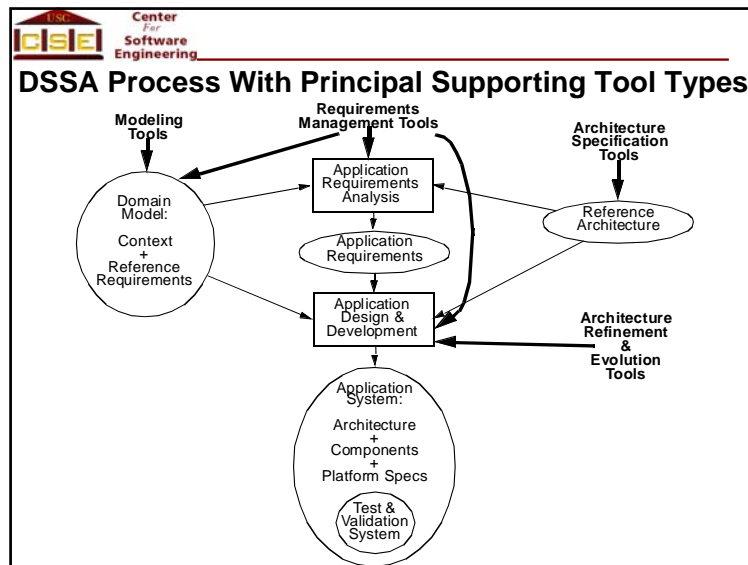
15



Ways To Specify Reference Architecture

<input type="checkbox"/> Architectural style <ul style="list-style-type: none">– Too general– No domain knowledge	<input type="checkbox"/> Parametric architecture <ul style="list-style-type: none">– Predefined variations– Easy instantiation & analysis– Limited variability
<input type="checkbox"/> Under-constrained architecture <ul style="list-style-type: none">– Deliberate incompleteness<ul style="list-style-type: none">• Difficult to analyze– Applicable to “large core + extensions” model	<input type="checkbox"/> Service-oriented architecture <ul style="list-style-type: none">– Collection of services with specified dependencies– Functionality-oriented
<input type="checkbox"/> Variance-free generic architecture	

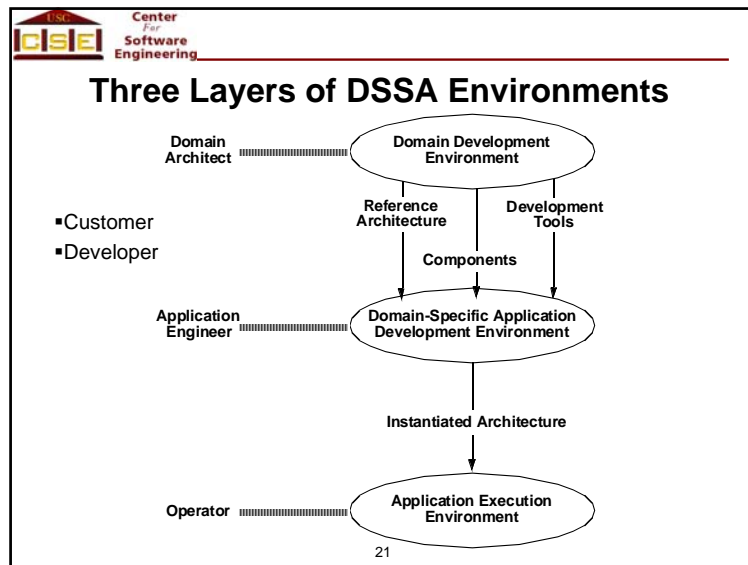
16



- ### Primary DSSA Support Tool Types
- Domain modeling tools
 - Requirements management tools
 - Describe new application objects, attributes, & relationships
 - Detect inconsistency, incompleteness, ambiguity
 - Architecture specification tools
 - Refine reference architectures
 - Describe & constrain components & connectors
 - Architecture refinement & evolution tools
 - Configure reference architecture to meet application-specific requirements
 - Specify & instantiate components
 - Compose configurations into executable form
- 18

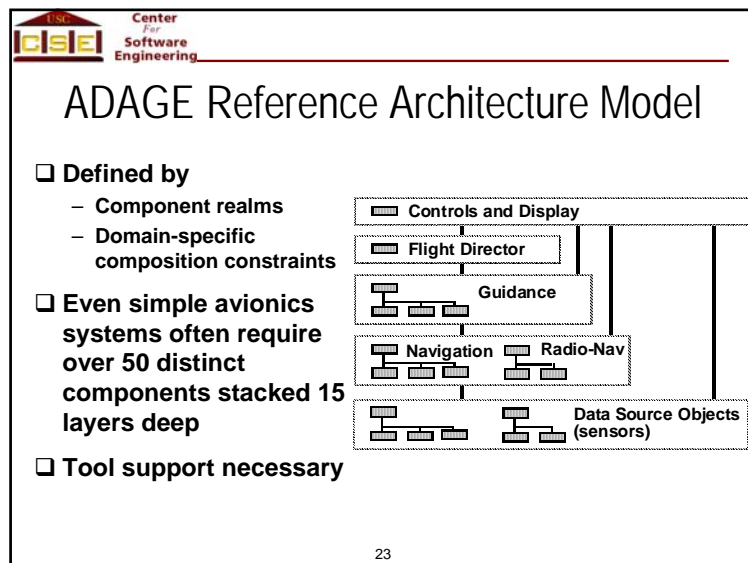
- ### Secondary DSSA Support Tool Types
- Repository tools
 - Store & retrieve components
 - Component selection tools
 - Component generators
 - Architecture package, load, & exercise tools
 - Aid integration & configuration of components
 - Requirements validation tools
 - Testing & analysis
 - Performance validation tools
 - Observe system in execution or simulation
- 19

- ### Tertiary DSSA Support Tool Types
- Language processing & compiling
 - User interfaces
 - Databases
 - Distributed and real-time computing
 - Documentation
 - Configuration management
- Generic rather than DSSA-specific
 - Used to develop DSSA-specific tools
- 20



Avionics DSSA

- Avionics Domain Application Generation Environment (ADAGE)
 - Layered reference architecture
 - Subsystems decomposed into primitive components with standardized interfaces
 - Over 40 different realms with over 350 distinct components

$$\text{realm} \equiv \{ x : \text{component} \mid (\forall i,j) (x_i.\text{interface} = x_j.\text{interface}) \}$$


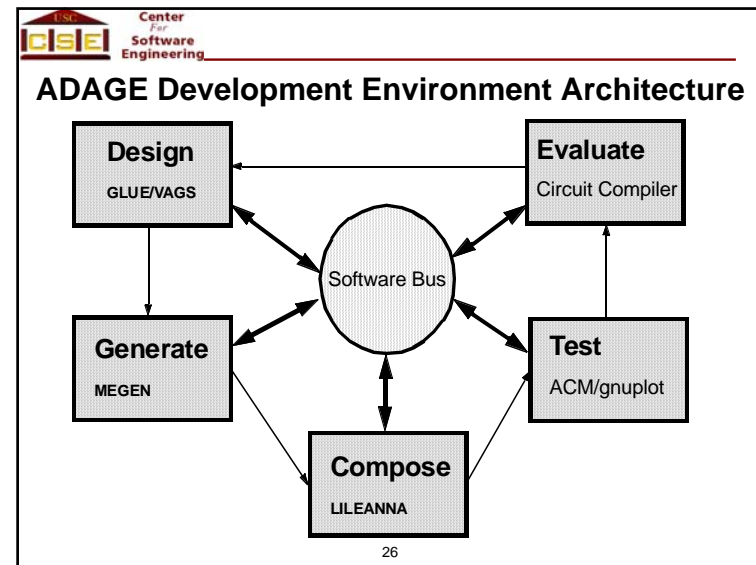
ADAGE Tool Suite

- DOMAIN (domain models, all integrated)
 - Hypermedia scenario-based requirements capture & modeling tool
- Avionics example test bed
 - Simulation & analysis tools
- Software circuit compiler
 - Timing analysis tool
- Realm component builder and compiler
 - Tool for describing components in the reference architecture
- GLUE (graphical layout user environment)
 - GUI for configuring and viewing the reference architecture
- VAGS (Variational attribute grammar system)
 - Constraint checker

ADAGE Tool Suite

- MEGEN** (Module Expression GENERator)
 - Application generator
- LILEANNA**
 - Ada-specific high-level system description language
- Scientist Work Bench**
 - Data/program organization/launching tool
- Software Component Design Record Editor**
 - Information store for organizing reusable software artifacts

25



Signal Processing DSSA

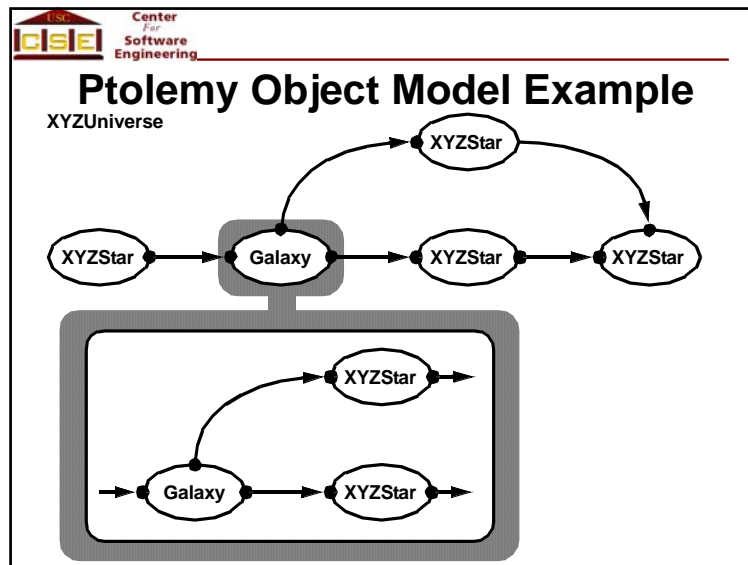
- Ptolemy**
 - Framework for simulation & rapid prototyping of heterogeneous systems
 - Focus on signal processing and communication systems
- Comprehensive scope**
 - Algorithms & communication strategies
 - Simulation
 - Hardware & software design
 - Parallel computing
 - Generating real-time prototypes
- Domain (new telecom services) requires joint design of**
 - Control software
 - Signal processing
 - Transport
 - Hardware elements

27

Ptolemy Object Model

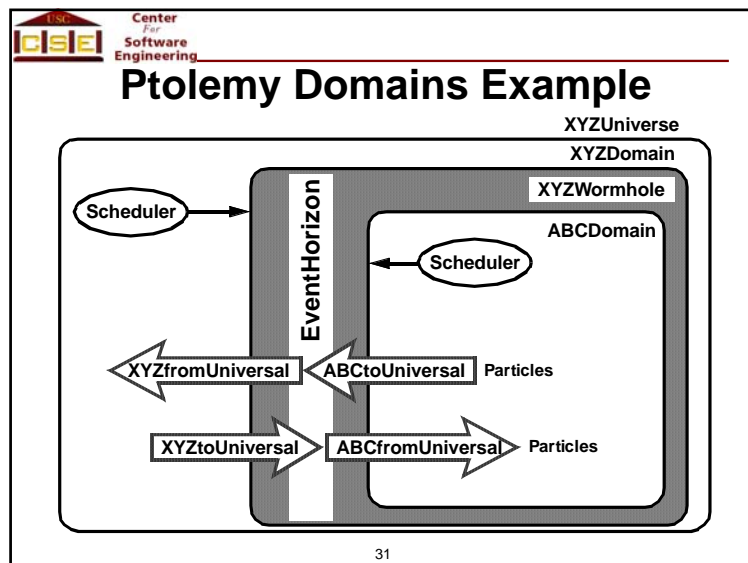
- Block* — reusable S/W component
- Star* — atomic Block
- Galaxy* — composite Block
- Universe* — complete Ptolemy application
- PortHoles* — standard interface for Block communication
- Particles* — Blocks communicate using streams of these
- Scheduler* — determines operational semantics of block network by ordering their invocations
- Target* — controls execution
 - For simulation-oriented application
 - Invokes Scheduler
 - For synthesis-oriented applications,
 - Synthesizes code
 - Invoke assembler
 - Downloads code into attached hardware, etc.

28



Ptolemy Object Model (cont.)

- Domain
 - Set of blocks, targets, & associated schedulers conforming to common computational model
 - Computational model focuses on block interactions; not internal
 - Operational not denotation
 - Example domains:
 - SDF - synchronous dataflow; for signal processing
 - DDF - dynamic dataflow; for signal processing
 - MQ - message queue; for telecom switching software
 - DE - discrete event; network or high-level system modeling
- Wormhole
 - Mechanism enabling different domains to coexist in Ptolemy application
 - Externally:
 - Star that obeys operational semantics of external domain
 - Internally:
 - Entire foreign universe, with scheduler & stars for that domain



Summary

- Domain-Specific Software Architecture (DSSA) is
 - Generalized solution for particular type of problem (domain)
 - Domain model
 - Reference requirements
 - Reference (parameterized) architecture
 - Supporting infrastructure/environment
 - Process/methodology to instantiate, refine, & evaluate it
 - Configurable for specific problem
- Benefits
 - Efficient development
 - Don't need to start from scratch
 - Requirements
 - Design
 - Implementation
 - Reuse
 - Requirements
 - Components
 - Architectures/patterns